R E P U B L I Q U E F R A N C A I S

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

PCT/FR 00/01767
10/018780
#2/Priority

REC'D 02 AUG 2000
WIPO PCT

8/6/02

BREVET D'INVENTION

EU

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 10 MAI 2000

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS Cédex 08
Téléphone : 01 53 04 53 04
Télécopie : 01 42 93 59 30

ETABLISSEMENT PUBLIC NATIONAL CREE PAR LA LOI N° 51-444 DU 19 AVRIL 1951

THIS PAGE BLANK (USPTO)

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 93 59 30

Code de la propriété intellectuelle-Livre VI

N° 55-1328

REQUÊTE EN DÉLIVRANCE

Confirmation d'un dépôt par télécopie ☐

Cet imprimé est à remplir à l'encre noire en lettres capitales

Réservé à l'INPI

DATE DE REMISE DES PIÈCES 25 JUIN 1999
N° D'ENREGISTREMENT NATIONAL 9908172
DÉPARTEMENT DE DÉPÔT 75 INPI PARIS
DATE DE DÉPÔT 25 JUIN 1999

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

CABINET NETTER
40 rue Vignon
75009 PARIS

n°du pouvoir permanent | références du correspondant | téléphone
INRIA Aff. 40 | 01 47 42 02 23

2 DEMANDE Nature du titre de propriété industrielle

☒ brevet d'invention ☐ demande divisionnaire ▷ demande initiale
☐ certificat d'utilité ☐ transformation d'une demande de brevet européen ▽ ☐ brevet d'invention ☐ certificat d'utilité n° date

Établissement du rapport de recherche ☐ différé ☒ immédiat

Le demandeur, personne physique, requiert le paiement échelonné de la redevance ☐ oui ☐ non

Titre de l'invention (200 caractères maximum)

Dispositif de gestion d'échanges de données entre matériels informatiques.

3 DEMANDEUR (S) n° SIREN . . . . . code APE-NAF . . .

Nom et prénoms (souligner le nom patronymique) ou dénomination

INRIA INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

L'UNIVERSITE PIERRE ET MARIE CURIE (Paris 6)

Forme juridique

Etablissement Public national à caractère scientifique et technologique.

Etablissement Public à Caractère Scientifique Culturel et Professionnel

Nationalité (s) française

Adresse (s) complète (s) Pays

Domaine de Voluceau
Rocquencourt - BP 105
78153 - LE CHESNAY CEDEX — France

4, Place Jussieu
75252 - Paris Cedex 05

En cas d'insuffisance de place, poursuivre sur papier libre ☐

4 INVENTEUR (S) Les inventeurs sont les demandeurs ☐ oui ☒ non Si la réponse est non, fournir une désignation séparée

5 RÉDUCTION DU TAUX DES REDEVANCES ☐ requise pour la 1ère fois ☐ requise antérieurement au dépôt : joindre copie de la décision d'admission

6 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE

pays d'origine | numéro | date de dépôt | nature de la demande

7 DIVISIONS antérieures à la présente demande n° date n° date

8 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (nom et qualité du signataire)

N° Conseil 92-1197 (B) (M)

SIGNATURE DU PRÉPOSÉ À LA RÉCEPTION | SIGNATURE APRÈS ENREGISTREMENT DE LA DEMANDE À L'INPI

La loi n°78-17 du 6 janvier 1978 relative à l'informatique aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION, CERTIFICAT D'UTILITE

DÉSIGNATION DE L'INVENTEUR
(si le demandeur n'est pas l'inventeur ou l'unique inventeur)

DEPARTEMENT DES BREVETS

26bis, rue de Saint-Pétersbourg
75800 Paris Cédex 08
Tél. : 01 53 04 53 04 - Télécopie : 01 42 93 59 30

N° D'ENREGISTREMENT NATIONAL

9908172

TITRE DE L'INVENTION :

Dispositif de gestion d'échanges de données entre matériels informatiques.

LE(S) SOUSSIGNÉ(S)

Mandataire
Cabinet NETTER
40 rue Vignon
75009 PARIS

DÉSIGNE(NT) EN TANT QU'INVENTEUR(S) (indiquer nom, prénoms, adresse et souligner le nom patronymique) :

- ROUILLIER Fabrice
8 rue des Bonnetiers
60200 COMPIEGNE

- FAUGERE Jean-Christophe
12 avenue Parmentier
75011 PARIS

NOTA : A titre exceptionnel, le nom de l'inventeur peut être suivi de celui de la société à laquelle il appartient (société d'appartenance) lorsque celle-ci est différente de la société déposante ou titulaire.

Date et signature (s) du (des) demandeur (s) ou du mandataire

Paris, le 25 juin 1999
N° Conseil 92-1197 (B) (M)

BA 113/140897

Dispositif de gestion d'échanges de données entre matériels informatiques

L'invention concerne le domaine de l'échange de données entre matériels informatiques de types identiques ou non, tels que des logiciels, des microprocesseurs, des bases de données, et analogues.

Dans le domaine informatique, l'utilisation de microprocesseurs de plus en plus puissants permet de réduire toujours plus les temps de calcul ou de traitement. Cependant, cela entraîne une manipulation, en temps réel, d'un nombre de données toujours plus important, par exemple supérieur à 1 Go (Giga-octets) dans le cas des systèmes tels que FGB et RS, ou dans le cas de systèmes de gestion de bases de données (SGBD) et de traitement d'images.

Dans un microprocesseur, ou plus généralement dans un matériel informatique, les données sont généralement stockées sous forme de "paquets" de k bits (information binaire) dans des registres d'une capacité de (n * k) bits. On entend ici par registre, les registres flottants, certains composants mémoire des microprocesseurs, les cartes graphiques, les MMX et analogues. Un microprocesseur ne sait donc pas traiter des données (entiers) dont la taille est supérieure à la capacité de ses registres, soit (n * k) bits.

En général, chaque "paquet" comprend k = 8 bits. On parle alors d'octet. Par exemple, le plus grand entier que peut traiter un microprocesseur 32 bits comprend 4 octets (n=4, k=8).

L'ordre dans lequel sont stockés les paquets de k bits (par exemple des octets) varie souvent d'une machine à l'autre. On dit alors que les matériels présentent des codages internes différents, ou en d'autres termes des arrangements de paquets de k bits différents.

Or, quelle que soit la machine, la représentation binaire d'un objet-donnée (par exemple un entier) de taille inférieure à k bits est invariante (tous les bits sont donnés dans le même ordre). Une telle représentation est donc commune à tout le monde.

Un réel problème se pose donc lorsque deux matériels fonctionnant selon des codages internes différents souhaitent échanger des objets-données (scalaires) dont les dimensions sont supérieures ou égales à k bits. Ce problème est encor renforcé lorsque les dimensions (k * n) des registres des matériels diffèrent.

A titre d'exemple, l'entier 33.751.553 qui se décompose dans la base $\{2^8\}$ sous la forme $1 + 2*2^8 + 3*2^{16} + 2*2^{24}$, est codé dans un microprocesseur de type ALPHA ou PC par la suite de n = 4 coefficients [1,2,3,2]. Or, dans un microprocesseur de type SPARC cet entier est codé par la suite de n = 4 coefficients [2,3,2,1] qui pour un microprocesseur désigne l'entier $16.909.058 (2 + 3*2^8 + 2*2^{16} + 2^{24})$.

Dans cet exemple, on s'aperçoit qu'une permutation des coefficients différencie les deux codages internes.

Pour permettre à de tels matériels d'échanger leurs entiers, il est donc indispensable qu'ils connaissent leurs codages internes respectifs, ou en d'autres termes les permutations qui leurs permettront de transformer leurs codages respectifs.

Or, actuellement, les permutations sont données en fonction de couples (k,n) fixés une fois pour toute, généralement à l'aide d'un logiciel tel que XDR (Marque déposée par la Société SUN).

Un tel logiciel assure en fait la transcription d'entiers machine et flottants codables sur 8, 16 et 32 bits (une convention est proposée pour les entiers de 64, mais pas au-delà) en un codage externe (ou codage de transmission) qui se

trouve être identique au codage interne des microprocesseurs de type SPARC. Le codage externe peut être qualifié de "langage commun ou universel". Dans ce logiciel, l'arrangement des k bits de chaque paquet (octet pour k=8) est toujours invariant.

Ce type de transcription nécessite, pour chaque échange de données, une première conversion (ou encodage) du premier code interne du matériel "émetteur" vers le codage externe, puis une seconde conversion (ou décodage) du code extern vers le second code interne du matériel récepteur.

La double conversion s'effectue également lorsque les matériels sont compatibles entre eux (même codage interne et incompatible avec un codage externe de type XDR). Pour éviter cela il est bien entendu possible de reconfigurer le logiciel XDR, mais cela impose une manipulation par un opérateur.

Par ailleurs, en l'état, le logiciel XDR est difficile à utiliser dans les environnements de 64 bits, et inutilisable dans les environnements de 128 bits. Plus généralement, dès que des données scalaires dépassent 32 bits, le logiciel XDR laisse la direction des opérations ("la main") à l'utilisateur pour traiter les données supérieures à 32 bits. D'autre part, il n'est pas prévu pour fonctionner avec des paquets différents de 8 bits.

Les solutions connues ne permettent donc pas une paramétrisation dynamique et/ou des échanges entre matériels indépendamment de leurs architectures respectives.

En résumé, aucune solution connue n'apporte une entière satisfaction en matières de rapidité, d'efficacité et d'adaptabilité.

L'invention a donc pour but d'améliorer la situation en matière d'échange de données.

α

Elle propose à cet effet un dispositif destiné à travailler sur des données élémentaires primaires codées individuellement selon un premier arrangement de mots (ou codage interne), et comprenant :

* des moyens de mémorisation où se trouvent stockés des premier et second jeux de symboles, tous distincts, formant respectivement une représentation du premier arrangement et d'un second arrangement de mots (ou codage externe), à priori distinct du premier, et
* un opérateur capable de recevoir en entrée les premier et second jeux de symboles et une donnée élémentaire primaire, tel qu'un entier, pour effectuer sur celle-ci des transformations de mots définies uniquement par les premier et second jeux de symboles, de manière à fournir en sortie une donnée secondaire correspondante et équivalente à la donnée élémentaire primaire.

On dispose ainsi d'un convertisseur entièrement paramétrable et dynamique.

L'invention trouve une application particulièrement intéressante lorsqu'un premier et un second matériel souhaitent échanger des données élémentaires primaires. Dans ce cas, le premier matériel délivre des données élémentaires primaires codées selon le premier arrangement (ou premier codage interne), tandis que des données élémentaires primaires codées selon un quatrième arrangement (second codage interne) par un second matériel sont converties par un moyen de conversion sous forme de données secondaires codées selon un troisième arrangement (second codage externe).

Comme indiqué en introduction, le mot arrangement doit être ici considéré en tant qu'agencement de groupes de bits dans un registre (en pratique, chaque groupe étant généralement formé de 8 bits (ou octet)).

Selon l'invention, l'opérateur du dispositif comprend des moyens d'interrogation qui effectuent les opérations suivantes :

* tout d'abord, ils fournissent au second matériel un message qui contient le second jeu de symboles et requiert l'envoi, en retour, d'une donnée élémentaire primaire, transformée du second jeu de symboles par le codage selon le quatrième arrangement;
* puis, ils déduisent de cette donnée élémentaire primaire et des premier et second jeux de symboles un troisième jeu de symboles formant une représentation du quatrième arrangement;
* ensuite, ils remplacent le second jeu de symboles par le troisième jeu de symboles, à la fois dans l'opérateur et dans le moyen de conversion, de sorte que :
  - en cas d'émission d'une donnée élémentaire primaire codée selon le premier arrangement et destinée au second matériel, l'opérateur lui délivre, directement, une donnée élémentaire primaire codée selon le quatrième arrangement, et
  - en cas d'émission d'une donnée élémentaire primaire codée selon le quatrième arrangement et destinée au premier matériel, l'opérateur lui délivre, directement, une donnée élémentaire primaire codée selon le premier arrangement.

De la sorte, notamment lorsque les matériels sont de types radicalement différents, et par conséquent présentent des codages internes (premier et quatrième arrangements) et externes (second et troisième arrangements) différents, le dispositif selon l'invention peut être configuré indépendamment des architectures des matériels souhaitant échanger des données.

Une fois que le dispositif a établi une liaison directe (c'est à dire à réalisé un "opérateur de conversion directe" entre les premier et second codages internes), le temps nécessaire à l'échange de données entre les matériels est très notablement réduit.

L'invention concerne également les procédés qui seront décrits ci-après et qui permettent au dispositif d'assurer leurs conversions.

D'autres caractéristiques et avantages de l'invention apparaîtront à l'examen de la description détaillée ci-après, et des dessins annexés, sur lesquels :

- la figure 1 est un schéma illustrant un mode de réalisation de l'invention dans une application à l'échange de donné s entre deux ordinateurs (ou plus généralement deux matériels informatiques) de types différents; et

- la figure 2 est un algorithme décrivant un mode de réalisation du procédé selon l'invention.

Les dessins annexés sont, pour l'essentiel, de caractère certain. En conséquence, ils pourront non seulement servir à compléter celle-ci, mais aussi contribuer à la définition de l'invention le cas échéant.

On se réfère tout d'abord à la figure 1 pour dresser un état des lieux de ce qui se faisait avant la présente invention en matière d'échange de données entre deux matériels informatiques, tels que des ordinateurs (ou station de travail) M1 et M2.

Bien entendu, il pourrait s'agir, plus simplement, de microprocesseurs, ou même de logiciels différents ou de bases de données, éventuellement implantés dans une même machine (ou ordinateur), mais fonctionnant selon des codages internes et/ou externes différents.

Il est important de noter que la figure 1 ne représente pas l'art antérieur en tant que tel, mais qu'elle permet de présenter les éléments

Dans l'exemple illustré sur la figure 1, l'ordinateur M1 comporte un microprocesseur 1, par exemple de type SPARC à 32 bits. Ce microprocesseur 1 est installé sur une carte électronique pour pouvoir coopérer avec un disque dur 2.

α

Le microprocesseur 1, sous le contrôle du système d'exploitation 3 de l'ordinateur, stocké sur le disque dur 2, effectue des opérations sur des données, et délivre sur une sortie 4 des données de 32 bits (lorsqu'il est de type SPARC) selon un premier arrangement (ou premier codage interne).

Chaque donnée de 32 bits est alors délivrée en sortie 4 du microprocesseur 1 sous la forme d'une suite ordonnée de quatre octets (8 bits). On appelle code interne $M1_{k,n}(E)$ la suite ordonnée de quatre (n) octets (k=8) représentant un entier E dans un microprocesseur SPARC 32 bits. Avec un tel microprocesseur SPARC, le codage interne d'un entier de 32 bits est la suite des coefficients de sa décomposition en base $\{2^k\}$, ordonnée selon les puissances décroissantes.

A titre d'exemple, dans un microprocesseur PC ou ALPHA, le codage interne d'un entier de 32 bits est la suite des coefficients de sa décomposition en base $\{2^8\}$, ordonnée selon les puissances croissantes.

Pour permettre la transmission de tels entiers E, de l'ordinateur M1 à l'ordinateur M2, il est nécessaire que les entiers présentent un même format, ou codage externe. Tel n'est pas toujours le cas, comme on le verra plus loin.

En conséquence, on prévoit, classiquement, un module de conversion 5, généralement implanté sous la forme d'un logiciel (ou programme) sur le disque dur de chaque ordinateur. Bien entendu, le module de conversion peut être réalisé sous la forme d'un circuit électronique.

Quoiqu'il en soit, ce module de conversion 5 est relié à une interface 6 couplée, par exemple par une liaison filaire, à l'interface 6' de l'ordinateur M2 avec lequel il souhaite échanger des données.

Pour définir le codage externe (ou second arrangement), on fait appel à une base $\{1, 2^k, 2^{2k}, \ldots 2^{nk}\}$, où k et n désignent respectivement le nombre de bits de chaque mot de

α

l'arrangement et le nombre d'éléments de la base, et où n*k est égal au nombre de bits de la donnée.

Par exemple, dans le cas précité, le premier arrangement est défini par n = 4 mots de k = 8 bits, soit n*k = 32 bits.

Dans cet exemple, le module de conversion 5 a donc pour fonction de convertir une donnée élémentaire primaire délivrée par la sortie 4 du microprocesseur 1 (c'est-à-dire fournie selon le premier codage interne ou premier arrangement $M_{1,k,n}$) en une donnée secondaire codée selon le codage externe, ou second arrangement, $D_{1,k,n}$.

Généralement, ce qui différencie le codage interne du codage externe, dans un même matériel, c'est une opération de type permutation. Dans ce cas, on a la relation :
$D_{1,k,n}(E) = \Phi_{1,k,n}\ (M_{1,k,n}(E))$

Il est bien évident, que dans certains cas, la permutation $\Phi_{m,k,n}$ (m=1,2) peut être l'identité. Dans ce cas, M et D sont constitués de la même suite ordonnée de mots définissant l'entier. On dit alors qu'ils présentent le même arrangement, ou format.

Ce qui vient d'être dit pour le premier ordinateur M1 s'applique également au second ordinateur M2. Seuls le codage interne $M_{2,k,n}$, ainsi qu'éventuellement le codage externe $D_{2,k,n}$, sont différents. On entend ici par différents, soit des arrangements (ou suites) dont les éléments (ou mots) sont ordonnés de façon différente, soit des arrangements qui ne présentent pas le même nombre d'éléments (k1 et k2 différents et/ou n1 et n2 différents).

Généralement, notamment lorsque l'on se trouve dans un environnement de type client-serveur, les matériels sont sensiblement homogènes, si bien que les codages externes qu'ils utilisent sont identiques. Dans ce cas, on a la relation suivante :

α

$$\Phi_{2,k,n}(M_{2,k,n}(E)) = D_{k,n}(E) = \Phi_{1,k,n}(M_{1,k,n}(E))$$

Dans les machines classiques, pour chaque donnée élémentaire primaire délivrée par le microprocesseur 1, sur sa sortie 4, et codée selon le premier arrangement (ou premier codage interne), le module de conversion 5 effectue un encodage (ou première conversion) destiné à fournir à l'interface 6 une donnée secondaire (généralement un entier E) codée selon le second arrangement (ou premier codage externe). En d'autres termes, en sortie du moyen de conversion 5 on dispose de la donnée suivante :

$$D_{k,n,}(E) = \Phi_{1,k,n}(M_{1,k,n}(E))$$

Cette donnée secondaire est alors adressée au second ordinateur M2, qui va la décoder (seconde conversion) à l'aide de son module de conversion 5' ($M_{2,k,n}(E) = \Phi^{-1}{}_{2,k,n}(D_{k,n,}(E))$), pour fournir au microprocesseur 1' une donnée élémentaire primaire codée selon son second codage interne (ou quatrième arrangement), de sorte qu'il puisse traiter cette donnée.

Il en va de même lorsque le second ordinateur M2 souhaite transmettre une donnée élémentaire primaire au premier ordinateur M1, et notamment à son microprocesseur 1. L'encodage consiste alors à former la donnée secondaire :

$$D_{2,k,n}(E) = \Phi_{2,k,n}(M_{2,k,n}(E))$$

et le décodage de la donnée secondaire fournie par le second ordinateur M2 s'effectue dans le module de conversion 5 du premier ordinateur M1. Ce qui fournit une donnée élémentaire primaire selon le premier arrangement (ou premier codage interne) : $M_{1,k,n}(E) = \Phi^{-1}{}_{1,k,n}(D_{k,n,}(E))$

Le module de conversion 5 de chaque matériel Mi (i=1,2) ne peut effectuer qu'un seul et unique codage du codage interne vers le codage externe, et réciproquement. Il en résulte que ce type de matériel ne peut fonctionner qu'avec des matériels présentant au moins un codage externe $D_{k,n}$ commun.

α

L'homme de l'art a proposé d'implanter dans certains matériels, par exemple sur leur disque dur, un logiciel d'échange de données. On citera, par exemple, le logiciel XDR (Marque déposée) du fabricant de stations de travail SUN. Ce logiciel propose une bibliothèque de fonctions d'encodage/décodage qui permettent à un premier matériel d'un premier type, d'échanger des données avec un second matériel d'un second type, une fois que les types respectifs de ces matériels ont été déclarés. Il s'agit, par conséquent, d'une opération de conversion de type purement statique, puisqu'elle requiert l'intervention d'un opérateur connaissant les types respectifs des deux matériels.

En outre, ce type de logiciel effectue systématiquement, pour chaque donnée à échanger, une double conversion. La première conversion consiste en l'encodage de la donnée élémentaire primaire selon le premier codage interne en une donnée secondaire selon le codage externe. La seconde conversion consiste en un décodage de la donnée secondaire selon le codage externe en une donnée élémentaire primaire selon le second codage interne.

Par ailleurs, à moins d'être reconfiguré à la main, par un spécialiste, le logiciel XDR continue d'effectuer sa double conversion lorsque les deux matériels sont identiques.

Cette double conversion ralentit énormément (au moins d'un facteur deux (2) les vitesses de traitement des données.

L'invention vient apporter une solution à cet inconvénient.

Dans ce qui suit, l'invention va être décrite, en référence aux figures 1 et 2, dans une application à l'échange de données entre deux matériels informatiques, tels que des ordinateurs (ou station de travail) M1 et M2.

Bien entendu, il pourrait s'agir, plus simplement, de microprocesseurs, ou même de logiciels différents, éventuel-

α

lement implantés dans une même machine (ou ordinateur), mais fonctionnant selon des codages internes différents.

L'invention propose un dispositif comprenant une première partie qui remplace le module de conversion 5 dans le premier matériel M1, et une seconde partie qui complète la première et est implantée au moins partiellement dans le premier matériel M1 (le reste étant alors implanté dans le second matériel M2).

De préférence, le dispositif est réalisé sous la forme de modules logiciels implantés sur le(s) disque(s) dur(s). Mais, il peut également être réalisé sous la forme de circuits électroniques. Une combinaison des deux (logiciel et circuit) peut également être envisagée.

Dans le premier ordinateur M1, se trouvent implantés des moyens de mémorisation capables de stocker un premier jeu (ou suite) de symboles, tous distincts, formant une représentation du premier arrangement (ou premier codage interne). De tels moyens de mémorisation sont, par exemple, réalisés sous la forme de lignes de programmes qui renvoient à des adresses de registres ou mémoires du disque dur 2 de l'ordinateur M1.

Les moyens de mémorisation 7 stockent également un second jeu (ou suite) de symboles, tous distincts, formant une représentation du second arrangement de mots (ou premier codage externe), généralement distinct du premier arrangement. Il peut en effet être identique dans certains cas.

De préférence, les jeux de symboles sont constitués d'une suite ordonnée de n composantes qui caractérisent, comme indiqué ci-avant, les premier et second arrangements.

Le dispositif comprend en outre, un opérateur 8 couplé aux moyens de mémorisation 7, ainsi qu'à la sortie 4 du microprocesseur 1. Il peut ainsi recevoir sur une entrée les premier et second jeux de symboles, ainsi que chaque donnée élémen-

α

taire primaire codée selon le premier arrangement par le microprocesseur 1.

Cet opérateur 8 est de préférence réalisé sous la forme d'un module logiciel (ou programme) faisant appel à une bibliothèque de calculs mathématiques. Il a pour fonction d'effectuer sur la donnée élémentaire primaire, reçue sous la forme d'une suite ordonnée de mots, des transformations de mots définies, uniquement, à partir des premier et second jeux de symboles. En sortie de cet opérateur 8, est délivrée une donnée secondaire équivalente à la donnée élémentaire primaire reçue. Le mot transformation doit être compris, ici, dans sa définition mathématique, c'est-à-dire en tant que fonction ou application.

De la sorte, on réalise un module de conversion perfectionné totalement configurable, et adaptable à tout type de matériel.

Le dispositif selon l'invention permet en outre d'accélérer de manière très sensible la vitesse d'échange de données entre deux matériels M1 et M2 de types différents, notamment. Dans ce qui suit, on considérera que non seulement les codages internes des deux matériels M1 et M2 sont différents, mais que leurs codages externes sont également différents. Par exemple, le microprocesseur 1 de l'ordinateur M1 est de type SPARC à 32 bits, tandis que le microprocesseur 1' du second ordinateur M2 est de type ALPHA à 64 bits.

Pour permettre à ces deux matériels M1 et M2 de communiquer, le dispositif selon l'invention met en oeuvre un protocole d'interrogation. Ce protocole est initié par le premier ordinateur M1 dans le but de déterminer le codage interne (ou quatrième arrangement) du microprocesseur 1' du second ordinateur M2.

L'interrogation est réalisée par un module d'interrogation 9, qui est constitué par un module logiciel (ou programme). Ce module logiciel d'interrogation est, de préférence, implanté

pour une partie 9-1 sur le disque dur de l'ordinateur M1, et pour une autre partie 9-2 complémentaire sur le disque dur 2' de l'ordinateur M2. Bien entendu, comme indiqué précédemment, chaque partie du module d'interrogation 9-1 et 9-2 pourra être réalisée sous la forme de circuits électroniques.

Par ailleurs, on peut envisager une variante dans laquelle l'intégralité du module d'interrogation 9 se trouve implantée sur le premier ordinateur M1. Dans ce cas, le module logiciel d'interrogation 9 est conçu de manière à implanter spontanément, lors d'une première interrogation, quelques lignes choisies de programme (sensiblement équivalentes à 9-2) sur le disque dur 2' du second ordinateur M2, et de préférence dans son module de conversion 5', lorsque celui-ci est réalisé sous la forme d'un module logiciel.

On décrit maintenant un mode (ou procédé) de fonctionnement du dispositif selon l'invention dans lequel le codage externe $D_{k,n}$ est supposé fixé à l'avance et identique pour les deux ordinateurs. Ce mode permet d'émuler le mode de fonctionnement du logiciel XDR.

Selon l'invention, le jeu de seconds symboles (représentatifs du second arrangement) est une suite ordonnée composée d'éléments distincts les uns des autres : [1, 2, 3, ..., n] et égale à $D_{k,n}(E)$. Cette suite ordonnée est associée à l'entier $E = 1 + 2*2^k + 3*2^{2k} + \ldots + n*2^{(n-1)*k}$, qui va être adressée au second ordinateur M2 pour déterminer son quatrième arrangement.

Pour des raisons de commodité, on suppose ici que n est inférieur à 2^k, le cas général pouvant être facilement traité en composant les procédés (ou modes de fonctionnement).

En d'autres termes, en adressant l'entier E indiqué ci-dessus, ou plus exactement en adressant la suite ordonnée [1, 2, 3, ..., n] comportant des nombres tous distincts les uns des autres, on va, en retour, en déduire le second code interne (ou quatrième arrangement) de la seconde machine M2.

A titre d'exemple, on prend k = 8 et k*n ≥ 32.

Pour déterminer complètement la permutation $\Phi_{2,k,n}$ mise en oeuvre par le module de conversion 5' du second ordinateur M2, il suffit de prendre l'entier $E = 1 + 2*2^8 + 3*2^{16} + 4*2^{24}$ et la donnée secondaire $D_{k,n}(E)$ formée de la suite de symboles [1, 2, 3, 4].

En pratique, la détermination de la permutation $\Phi_{2,k,n}$ revient à construire un tableau noté, pour k fixé, permut, tel que permut[n] soit la liste ordonnée représentant $\Phi_{m,k,n}$.

Le programme de détection du codage interne et de la construction du codage externe est donné à titre d'exemple, en langage C, dans le module Mod2 de l'annexe.

Les fonctions construisent des listes ordonnées external_coding[i] et internal_coding[i], pour des valeurs de i prenant les tailles des entiers usuels du langage C (char, short, int, et long). Il est rappelé qu'en langage C (qui n'est qu'un exemple de langage de programmation utilisable) la variable long indique la taille maximale des registres qu'utilise le microprocesseur pour stocker les données scalaires). Les variables external_coding et internal_coding représentent respectivement le codage externe $D_{k,n}$ et le codage interne $M_{k,n}$. Ces variables sont données à titre d'exemple dans le module Mod1 de l'annexe, où "B8" est le type invariant qui permet de représenter un entier de la valeur 0 (zéro) à la valeur 255.

Pour une valeur i fixée, on construit l'entier $E = 1 + 2*2^k + \ldots + n*2^{k*(i-1)}$.

Les listes ordonnées précitées forment des tableaux. Le tableau external_coding[i] est alors égal, pour k = 8, au codage externe $D_{8,i}(E)$ = [1, 2, 3, ..., i]. Le tableau internal_coding[i] est, quant à lui, égal à $M_{m,8,i}(E)$ qui varie selon l'ordinateur (ou machine considérée).

a

On définit ensuite un ensemble (ou multiplicité) de plusieurs fonctions de permutation $\Phi_{m,8,i}$ pour des valeurs de i variant de 1 à sizeof(long) (ou dans certains cas de 1 à siz of(longlong). Cette dernière variable définit le scalaire entier présentant la plus grande taille admissible dans le langage C; en générale il s'agit du plus grand entier compréhensible par la machine ou matériel. Ces fonctions de permutation assurent une prise en compte d'entiers codables sur 8, 16, 32 et 64 bits. Elles peuvent être aisément étendues à des valeurs plus grandes, notamment 128 bits.

Les fonctions de permutation sont définies, à titre d'exemple, dans le module Mod3 de l'annexe. On notera, que la formulation de ce module de programme permet de détecter les valeurs de i pour lesquelles aucune permutation n'est nécessaire.

Il faut ensuite décrire des fonctions d'encodage/décodage qui vont être utilisées lorsque les paramètres du problème, à savoir les codage interne, codage externe et fonction de permutation auront été déterminés.

Par exemple, la fonction permettant de transformer le codage externe d'un entier (ou tableau de m éléments de 8 bits) de taille donnée (sz) en son format interne peut être définie par le module Mod4 de l'annexe.

Comme le remarquera l'homme de l'art, le module Mod4 décrivant les fonctions d'encodage/décodage propose des fonctions génériques, dans la mesure où, d'une part, elles ne dépendent que de la taille des entiers à traiter, et que d'autre part, on prend soin de ne faire des permutations que si cela s'avère nécessaire.

De la même manière, on définit les fonctions inverses qui permettent de transformer le codage interne d'un enti r (tableau m d'élément de 8 bits) de taille donnée (sz) en un autre codage interne (tableau p d'éléments de 8 bits). Ces fonctions inverses sont définies, à titre d'exemple, par le

α

module Mod5 de l'annexe. Elles permettent donc de convertir directement une donnée élémentaire primaire codée selon le premier, respectivement quatrième, arrangement en une donnée élémentaire primaire codée selon le quatrième, respectivement premier, arrangement.

A l'aide des modules Mod1 à Mod5, on réalise un protocole (ou procédé) d'échange de données binaires dont l'implantation ne dépend en aucune manière des architectures respectives des ordinateurs (ou machines) considérés. Bien entendu, cette implantation est liée au langage utilisé, ici le langage C. Mais, une transposition vers un autre langage informatique peut être aisément obtenue.

Pour faire fonctionner le protocole d'échanges, le premier ordinateur M1 ouvre le canal de communication qui le relie au second ordinateur M2, puis active ce second ordinateur M2 et lui envoie l'entier E défini par le second jeu de symboles, ici [1, 2, 3, 4], sur le canal de communication choisi. Le second ordinateur M2 lit l'entier, le transforme en le codant selon son quatrième arrangement (ou second codage interne) et retourne cette transformée sur le canal de communication.

Un exemple de module de programme permettant de réaliser les opérations citées ci-dessus, est donné dans le module Mod6 (pour ce qui concerne la partie 9-1 implantée dans le premier ordinateur M1) et dans le module Mod7 (pour ce qui concerne la partie 9-2 implantée dans le second ordinateur M2).

Bien entendu, et comme indiqué précédemment, la partie 9-2 destinée à activer le second ordinateur M2 peut être implantée à distance par le premier ordinateur M1. Pour ce faire, il suffit que le module d'interrogation 9 de M1 adresse une adaptation du module de programme proposé dans Mod7.

Dans le module Mod6, deux arguments sont appelés : un premier nom de machine (premier ordinateur) et un second nom de machine (second ordinateur) qui doit être activé par la première machine. Par ailleurs, dans Mod7, deux arguments

sont également appelés : un nom de machine (ou ordinateur) et un port.

Dans ces deux modules, les fonctions send_n (nb, bus, buf, n), respectivement read_n (nb, bus, buf, n), écrivent, respectivement lisent, un tableau buf de n*k bits (k est en pratique fixé à la valeur 8) constitué d'entiers machines de nb*k bits sur un canal noté bus.

Il est clair que ces fonctions vont dépendre du canal de communication choisi (mémoire partagée, fichiers, bases de données (SGBD), cartes graphiques, microprocesseur, formats de stockage d'image et/ou de sons (multimédia), sockets et analogues). De telles fonctions font appel (sous forme de boucles) aux fonctions conv_machine_2_prot_UI et reorder_UI, et gèrent les buffers d'entrée/sortie si nécessaire.

Dans ce qui précède, le codage externe $D_{k,n}$ était supposé fixé à l'avance. Il correspondait donc à un mode de réalisation du dispositif selon l'invention particulièrement bien adapté aux matériels pouvant communiquer entre eux en raison d'un même codage externe $D_{k,n}$.

Cependant, le dispositif selon l'invention peut être également adapté à l'échange de données entre des matériels présentant des codages externes différents. Dans ce cas, le dispositif calcule de façon dynamique un codage externe commun, ce qui revient à ne pas fixer a priori de format d'échange des données.

Une solution préférentielle consiste à fixer comme codage externe d'échange le codage interne de l'une des deux machines (ou ordinateurs) de sorte que l'une des deux permutations $\Phi_{1,k,n}$ et $\Phi_{2,k',n'}$ utilisées par ces deux machines soit égale à l'identité.

Les principales étapes de détermination dynamique d'un format d'échange (ou codage externe) selon l'invention vont maintenant être décrites en référence à la figure 2.

α

Tout d'abord, dans une étape 10, le microprocesseur 1 du premier ordinateur M1 adresse au dispositif implanté, ici sur le disque dur 2, une donnée codée selon le premier arrangement (ou premier codage interne).

Cette donnée étant destinée au second ordinateur M2, dont on ne connaît ni le codage externe (ou troisième arrangement) ni le codage interne (ou quatrième arrangement), on effectue, dans une étape 20, une initialisation de la partie 9-1 du protocole contenue dans le premier ordinateur M1, et notamment dans l'opérateur 8. Cela consiste à fixer une valeur par défaut pour $D_{k,n}$. En fait, cette valeur par défaut pour $D_{k,n}$ est la suite ordonnée décrite précédemment [1, 2, 3, ..., n]. S'ensuit alors, dans une étape 30, une initialisation de la partie 9-2 du protocole qui se trouve dans le second ordinateur M2.

Il est clair, comme indiqué précédemment, que lorsque l'intégralité du programme d'interrogation se trouve implantée dans le premier ordinateur M1, les lignes de programme 9-2 assurant l'initialisation du second ordinateur M2 doivent être transférées à celui-ci, par exemple dans son module de conversion 5'. Cette étape 30 consiste donc à remplacer le codage externe $D_{2,k',n'}$ (ou troisième arrangement) du second ordinateur M2 par la valeur par défaut $D_{k,n}$ fixée à l'étape 20.

En fait, M1 adresse à M2 un message d'interrogation contenant le second jeu de symboles qui représente le second arrangement, ou bien une ou plusieurs variantes de celui-ci, comportant un (des) nombre(s) de symboles différent(s) (supérieur ou inférieur), de sorte qu'en cas de formats d'échange différents, l'un au moins de ces jeux de symboles puisse être traité par le module de conversion 5' de M2. Dans ce cas, ce sont les moyens de mémorisation 7 de M1 qui stockent l s différentes variantes de jeux de symboles comportant des nombres n_i de mots différents et/ou de mots de nombre k_i de bits différents.

d

Le microprocesseur 1' de M2 renvoie en retour au module de conversion 5', qui code désormais au format d'échange $D_{k,n}$ imposé par défaut, une donnée élémentaire primaire codée selon son quatrième arrangement (ou second codage externe) $M_{2,k',n'}$. Cette donnée est convertie (encodée) selon le codage externe $D_{k,n}$ et adressée à M1 sur le canal de communication.

Dans une étape 40, le module d'interrogation 9-1 implanté dans le premier ordinateur M1 va lire sur le canal de communication la donnée $M_{2,k',n'}$ (transformée du second jeu de symbole $D_{k,n}$) qui représente le codage interne du second ordinateur M2.

Dans une étape 50, le module d'interrogation 9-1 du premier ordinateur M1 remplace alors la valeur par défaut du codage externe $D_{k,n}$ par la donnée reçue du second ordinateur M2, à savoir $M_{2,k',n'}$.

Puis, dans une étape 60, on remplace dans le second ordinateur M2, et notamment dans son module de conversion 5', la valeur par défaut de $D_{k,n}$ fournie lors de l'étape 30.

On peut noter que, du fait du remplacement dans M2 de $D_{k,n}$ (codage externe imposé par défaut) par $M_{2,k',n'}$ (codage interne du microprocesseur 1'), on se retrouve dans le module de conversion 5' de M2 avec une permutation $\Phi_{2,k',n'}$ égale à l'identité. De même, on se retrouve dans le "module de conversion" 5 de M1 avec un codage externe identique au codage interne de M2, si bien que toute donnée élémentaire primaire d'un microprocesseur 1 ou 1', codée selon le premier ou le quatrième arrangement, peut être convertie directement en une donnée élémentaire primaire de l'autre microprocesseur 1' ou 1, codée selon le quatrième ou le premier arrangement.

Les deux machines M1 et M2 sont désormais prêtes à échanger directement des données élémentaires primaires dans une étape 70.

q

Une donnée élémentaire primaire délivrée par le microprocesseur 1' de M2, ou par le module de conversion 5 de M1, ne va donc pas subir de conversion dans le module de conversion 5'. Ce module de conversion 5' est, en quelque sorte, court-circuité. Par ailleurs, une donnée destinée au microprocesseur 1 de M1 est convertie, directement, du format interne (quatrième arrangement) de M2 au format interne (premier arrangement) de M1.

Il est clair, que le procédé qui vient d'être décrit en référence à l'algorithme illustré sur la figure 2 s'adapte automatiquement, par une phase de négociation, au cas, décrit précédemment, dans lequel les deux ordinateurs (ou machines) présentent d'origine le même codage externe. Dans ce cas, les étapes d'initialisation 20 et 30 sont inutiles.

Un exemple de programme, permettant de mettre en oeuvre l'algorithme illustré sur la figure 2, est donné dans le module Mod8 de l'annexe. A ce programme sont associés, comme indiqué précédemment, deux programmes, du type de ceux donnés dans les modules Mod6 et Mod7 de l'annexe. Il s'agit des programmes permettant respectivement d'activer le module de conversion 5 du premier ordinateur (module Mod9 de l'annexe) et du module permettant d'activer le second ordinateur M2 (module Mod10 de l'annexe).

Dans cette invention, la définition des données scalaires doit être prise dans son sens large, c'est-à-dire en tant qu'unités de base. Dans ces conditions, la notion de codage doit également être prise dans son sens le plus large, c'est-à-dire en tant que procédé (ou mode) destiné à ordonner des unités scalaires. Le codage pourra donc être forcé ou initié par un utilisateur, en tant que de besoins.

L'invention ne se limite pas aux modes de réalisation de dispositif et de procédé décrits ci-avant, seulement à titre d'exemple, mais elle englobe toutes les variantes que pourra envisager l'homme de l'art dans le cadre des revendications ci-après.

α

Ainsi, on a décrit un dispositif et le procédé associé dans lesquels le second jeu de symboles (représentatif du second arrangement) était constitué d'une suite de n éléments, tous distincts et de valeurs croissant de 1 à n. Mais, il est clair que toute autre suite de n éléments distincts pourra être utilisée.

Annexe

* Mod1 :

```
Typedef longlong Ilonglong
Typedef unsigned int UI32

#define _max_size_type 255
B8 external_coding[_max_size_type][_max_size_type];
B8 internal_coding[_max_size_type][_max_size_type];
UI32 permut[_max_size_type][_max_size_type];
```

* Mod2 :

```
void init_permut_char(String tab_int)
{
  tab_int[0]=1;
}

#define genere_permut(typ) \
void init_permut_ ## typ ## (String tab_int) \
  { \
     typ res=1; \
     typ tmp=2; \
     typ inc=256;\
     typ pow=1; \
     UI32 j=1; \
     for(j=1;j<sizeof(typ);j++) { \
        pow*=inc;\
        res+=(tmp)*pow;\
        tmp++; \
    }\
    memcpy((void *)tab_int,(void *)(&res),sizeof(typ));\
  }

genere_permut(short)
genere_permut(int)
genere_permut(long)
genere_permut(Ilonglong)
```

```
v id set_permutation_0(String int_c ding,const UI32 sz)
{
  if(sz==siz  f(char)){
    init_permut_char(int_coding);
  }
  if(sz==sizeof(short)){
    init_permut_short(int_coding);
  }
  if(sz==sizeof(int)){
    init_permut_int(int_coding);
  }
  if(sz==sizeof(long)){
    init_permut_long(int_coding);
  }
  if(sz==sizeof(Ilonglong)){
    init_permut_Ilonglong(int_coding);
  }
}

void set_permutations_0()
{
  UI32 i=1;
  while (i<=maxsize_prot) {
    set_permutation_0(&(internal_coding[i][0]),i);
    i*=2;
  }
}


/* maxsize_prot definit la taille maximale des entiers
   devant etre consideres par le protocole */

 void set_exchange_0()
 {
   UI32 i=1,j;
   while (i<=maxsize_prot) {
     for(j=0;j<i;j++){
        external_coding[i][j]=j+1;
     }
     i*=2;

   }
 }
```

* Mod3 :

```
UI32 find_elem(B8 elt,String tab,const UI32 sz)
{
  UI32 i=0;
  while ((i<sz) && ((tab[i])!=(elt))) i++;
  return(i);
}

void define_permutation(String int_coding,String ext_coding,
UI32 ** perm_coding,const UI32 sz)
{
  UI32 i;
  if(strncmp(int_coding,ext_coding,sz)){
    permut_type[sz]=NEED_PERMUT;
    for(i=0;i<sz;i++) {
      (*perm_coding)[i]=find_elem(int_coding[i],ext_coding,sz);
    }
  }
  else permut_type[sz]=NO_PERMUT;
}


void C__initProtocol()
{
  UI32 i=1;
  UI32 *ptr;
  set_exchange_0()
  C__initProtocol_0();
  while (i<=maxsize_prot) {
    ptr=&(permut[i][0]);
    define_permutation(&(internal_coding[i][0]),
                       external_coding[i],&ptr,i);
    i+=2;
  }
}
```

* Mod4 :

```
v id reorder_UI(const UI32 sz,String m)
{
  UI32 i;
  static B8 tmp[_max_size_type];
  UI32 *permut_tmp=NULL;
  if(permut_type[sz]==NEED_PERMUT) {
    permut_tmp=&(permut[sz][0]);
    for(i=0;i<sz;i++){
      tmp[i]=m[permut_tmp[i]];
    }
    for(i=0;i<sz;i++){
      m[i]=tmp[i];
    }
  }
}
```

* Mod5 :

```
void conv_machine_2_prot_UI(const UI32 sz,String p,Cste_String m)
{
  UI32 i;
  UI32 *permut_tmp=NULL;
  if(permut_type[sz]==NEED_PERMUT) {
    permut_tmp=&(permut[sz][0]);
    for(i=0;i<sz;i++){
      p[permut_tmp[i]]=m[i];
    }
  }
  else {
    for(i=0;i<sz;i++){
      p[i]=m[i];
    }
  }
}
```

* Mod6 :

```
*/
int main(int argc, char** argv)
{
  UI32 te=1234,re=0;
  /* ouvre le canal de communication et lance le serveur */
  BUS GB=createBus(argv[1],argv[2]);
  C_initProtocol();
  send_n(sizeof(UI32),GB,(char *)(&te),1);
  read_n(sizeof(UI32),GB,(char *)(&re),1);
  fprintf(stderr,''%u'',te);
  fprintf(stderr,''%u'',re);
}

/*
```

* Mod7 :

```
*/
int main(int argc, char** argv)
{
  UI32 te=0;
  /* ouvre le canal de communication et lance le serveur */
  BUS GB=createBus(argv[1],argv[2]);
  C__initProtocol();
  read_n(sizeof(UI32),GB,(char *)(&te),1);
  te++;
  send_n(sizeof(UI32),GB,(char *)(&te),1);
```

* Mod8 :

```
void C__reinitProtocol()
{
  UI32 i=1;
  UI32 *ptr;
  set_exchange_0()
  C__initProtocol_0();
  while (i<=maxsize_prot) {
    ptr=&(permut[i][0]);
```

* Mod9 :

```
*/
int main(int argc, char** argv)
{
  UI32 te=1234,re=0;
  /* taille maximale d'entiers supportee par ce processeur */
  B8 maxsize_loc=sizeof(long);
  /* taille maximale d'entiers supportee par le client */
  B8 maxsize_ext=0;
  /* ouvre le canal de communication et lance le serveur */
  BUS GB=createBus(argv[1],argv[2]);

  /*
     initialisation par defaut du protocole
  */
  C__initProtocol();

  read_n(1,GB,(char  )(&(maxsize_ xt)),1);
  send_n(1,GB,(char *)(&(maxsize_loc)),1);
  /*
     calcul d  la taille maximale d'entiers supportee
     par le protoc le
  */
```

α

```
maxsiz _prot=min(maxsize_l c,maxsiz _ext);

/*
   reception du codage interne pour la machine sur laquelle
   fonctionne le serveur des entiers admissible pour le
   protocole, affectation au codage d'echange
*/
for(i=1;i<=((UI32)maxsize_prot);i*=2){
  read_n(1,GB,(char *)(&(external_coding[i][0])),i);
}

/*
   re-initialisation du protocole avec le nouveau
   format d'echange
*/
C__reinitProtocol();

/* envoi des commandes */
send_n(sizeof(UI32),GB,(char *)(&te),1);

/* lecture du resultat */

read_n(sizeof(UI32),GB,(char *)(&re),1);
fprintf(stderr,''%u'',te);
fprintf(stderr,''%u'',re);
}
```

* Mod 10 :

```
*/
int main(int argc, char** argv)
{
  UI32 te=0,i;
  /* taille maximale d'entiers supportee par ce processeur */
  B8 maxsize_loc=sizeof(long);
  /* taille maximale d'entiers supportee par le client */
  B8 maxsize_ext=0;
  /* ouvre le canal de communication et lance le serveur */
  BUS GB=createBus(argv[1],argv[2]);

  /*
     initialisation par defaut du protocole
  */
  C__initPr t c l();

  send_n(1,GB,(char  )(&(maxsize_l c)),1);
  read_n(1,GB,(char  )(&(maxsize_ext)),1);
```

```
/*
   calcul de la taille maximale d'entiers supp rtee
   par le protocole
*/
maxsize_prot=min(maxsize_loc,maxsize_ext);

/*
   envoi du codage interne pour cette machine des entiers
   admissible pour le protocole
*/
for(i=1;i<=((UI32)maxsize_prot);i*=2){
  send_n(1,GB,(char *)(&(internal_coding[i][0])),i);
}

/*
   on affecte au format d'echange le format interne
   des entiers admissibles pour le protocole
*/

 for(i=1;i<=((UI32)maxsize_prot);i*=2) {
   for(j=0;j<i;j++) {
     external_coding[i][j]=internal_coding[i][j];
   }
 }

 /*
    re-initialisation du protocole avec le nouveau
    format d'echange
 */
 C__reinitProtocol();

 /* lecture des commandes */
 read_n(1,GB,(char *)(&(te)),1);
 te++;
 /* renvoit du resultat */
 send_n(sizeof(UI32),GB,(char *)(&te),1);
}
```

α

Revendications

1. Dispositif de conversion de données, destiné à travailler sur des données élémentaires primaires codées individuellement selon un premier arrangement de mots,

caractérisé en ce qu'il comprend :

* des moyens de mémorisation (7) pour stocker un premier jeu de symboles, tous distincts, formant une représentation dudit premier arrangement et un second jeu de symboles tous distincts, formant une représentation d'un second arrangement de mots, et

* un opérateur (8) agencé pour recevoir en entrée une donnée élémentaire primaire, ainsi que lesdits premier et second jeux de symboles, et pour effectuer sur cette donnée élémentaire primaire des transformations de mots définies uniquement par lesdits premier et second jeux de symboles de manière à fournir en sortie une donnée secondaire correspondante et équivalente à ladite donnée élémentaire primaire.

2. Dispositif selon la revendication 1, dans lequel un premier matériel (1) délivre lesdites données élémentaires primaires codées selon ledit premier arrangement, et un moyen de conversion (5') délivre des données secondaires codées selon un troisième arrangement, après conversion de données élémentaires primaires codées selon un quatrième arrangement par un second matériel (1'), lesdits matériels (1,1') désirant échanger des données élémentaires primaires,

caractérisé en ce que ledit opérateur (8) comprend des moyens d'interrogation (9) agencés pour :

* fournir audit second matériel (1') un message contenant ledit second jeu de symboles et requérant l'envoi en retour audit opérateur (8) d'une donnée élémentaire primaire, transformée dudit second jeu de symboles par codage selon ledit quatrième arrangement,

* déduire de cette donnée élémentaire primaire ainsi que des premier et second jeux de symboles un troisième jeu de symboles formant une représentation dudit quatrième arrangement,

d

* remplacer ledit second jeu de symboles par ledit troisième jeu de symboles, dans ledit opérateur (8) et dans ledit moyen de conversion (5'), de sorte qu'en cas d'émission d'une donnée élémentaire primaire codée selon le premier, respectivement quatrième, arrangement et destinée audit second, respectivement premier, matériel, ledit opérateur (8) délivre à celui-ci, directement, une donnée élémentaire primaire codée selon le quatrième, respectivement premier arrangement.

3. Dispositif selon la revendication 2, caractérisé en ce que lesdits premier, second et troisième jeux de symboles sont des suites ordonnées de nombres.

4. Dispositif selon la revendication 3, caractérisé en ce que ledit second jeu de symboles est la suite [1, 2, 3,..., n-1, n], n étant le nombre de composantes d'une base sur laquelle la donnée secondaire est décomposée en mots de k bits, k étant supérieur ou égal à 1, et en particulier égal à 8.

5. Dispositif selon l'une des revendications 2 à 4, caractérisé en ce que lesdits second et troisième arrangements sont identiques.

6. Dispositif selon l'une des revendications 2 à 4, caractérisé en ce que lesdits second et troisième arrangements sont différents et sont associés à des jeux de symboles comportant des nombres de mots différents et/ou des mots de nombre de bits différents, et en ce que ledits moyens d'interrogation sont agencés pour adresser audit moyen de conversion (5') au moins un second jeu de symboles de sorte qu'il soit substitué audit troisième arrangement.

7. Dispositif selon la revendication 6, caractérisé en ce que lesdits moyens de mémorisation (7) stockent plusieurs seconds jeux de symboles comportant des nombres n_i de mots différents et/ou de mots de nombre k_i de bits différents, et en ce que lesdits moyens d'interrogation (9) sont agencés

α

pour fournir audit second matériel (1'), par voie de message, un nombre choisi de premiers jeux de symboles différents.

8. Dispositif selon l'une des revendications 2 à 7, dans lequel ledit premier matériel (1) est implanté dans une première machine (M1), en particulier un ordinateur, caractérisé en ce que lesdits moyens de mémorisation (7) et une partie au moins dudit opérateur (8,9-1) sont implantés dans ladite première machine.

9. Dispositif selon la revendication 8, caractérisé en ce que lesdits moyens de mémorisation (7) et une partie (8, 9-1) au moins dudit opérateur sont implantés sous forme d'un programme dans ladite première machine (M1).

10. Dispositif selon l'une des revendications 8 et 9, dans lequel ledit second matériel (1') et ledit moyen de conversion (5) sont implantés dans une seconde machine (M2), en particulier un ordinateur, caractérisé en ce qu'une première partie (9-1) desdits moyens d'interrogation est implantée dans ladite première machine, tandis qu'une seconde partie (9-2) complémentaire est implantée dans ladite seconde machine (M2).

11. Dispositif selon la revendication 10, caractérisé en ce que lesdits moyens d'interrogation (9-2) sont implantés sous forme d'un programme.

12. Dispositif selon la revendication 11, caractérisé en ce que ledit opérateur (8) est agencé pour implanter ladite seconde partie (9-2) des moyens d'interrogation dans ladite seconde machine (M2) lorsque ledit premier matériel tente, pour la première fois, d'échanger des données élémentaires primaires avec ledit second matériel (1').

13. Procédé de conversion de données élémentaires primaires codées individuellement selon un premier arrangement de mots, caractérisé en ce qu'il comprend les étapes suivantes :

a) prévoir un premier jeu de symboles, tous distincts, formant une représentation dudit premier arrangement et un second jeu de symboles tous distincts, formant une représentation d'un second arrangement de mots, et

b) recevoir une donnée élémentaire primaire, ainsi que lesdits premier et second jeux de symboles, et

c) effectuer sur cette donnée élémentaire primaire des transformations de mots définies uniquement par lesdits premier et second jeux de symboles de manière à fournir en sortie une donnée secondaire correspondante et équivalente à ladite donnée élémentaire primaire.

14. Procédé selon la revendication 13, dans lequel on reçoit d'un premier matériel une donnée élémentaire primaire codée selon ledit premier arrangement, et d'un moyen de conversion une donnée secondaire codée selon un troisième arrangement, issue de la conversion d'une donnée élémentaire primaire codée selon un quatrième arrangement par un second matériel, caractérisé en ce qu'à l'étape b) :

* on fournit audit second matériel un message contenant ledit second jeu de symboles et requérant l'envoi en retour d'une donnée élémentaire primaire, transformée dudit premier jeu de symboles par codage selon ledit quatrième arrangement, puis

* on déduit de cette donnée élémentaire primaire ainsi que des premier et second jeux de symboles un troisième jeu de symboles formant une représentation dudit quatrième arrangement, et

* on remplace partout ledit second jeu de symboles par ledit troisième jeu de symboles, de sorte qu'en cas d'émission d'une donnée élémentaire primaire codée selon le premier, respectivement quatrième, arrangement et destinée audit second, respectivement premier, matériel, on délivre à celui-ci, directement, une donnée élémentaire primaire codée selon le quatrième, respectivement premier arrangement.

(32 pages)

CABINET NETTER

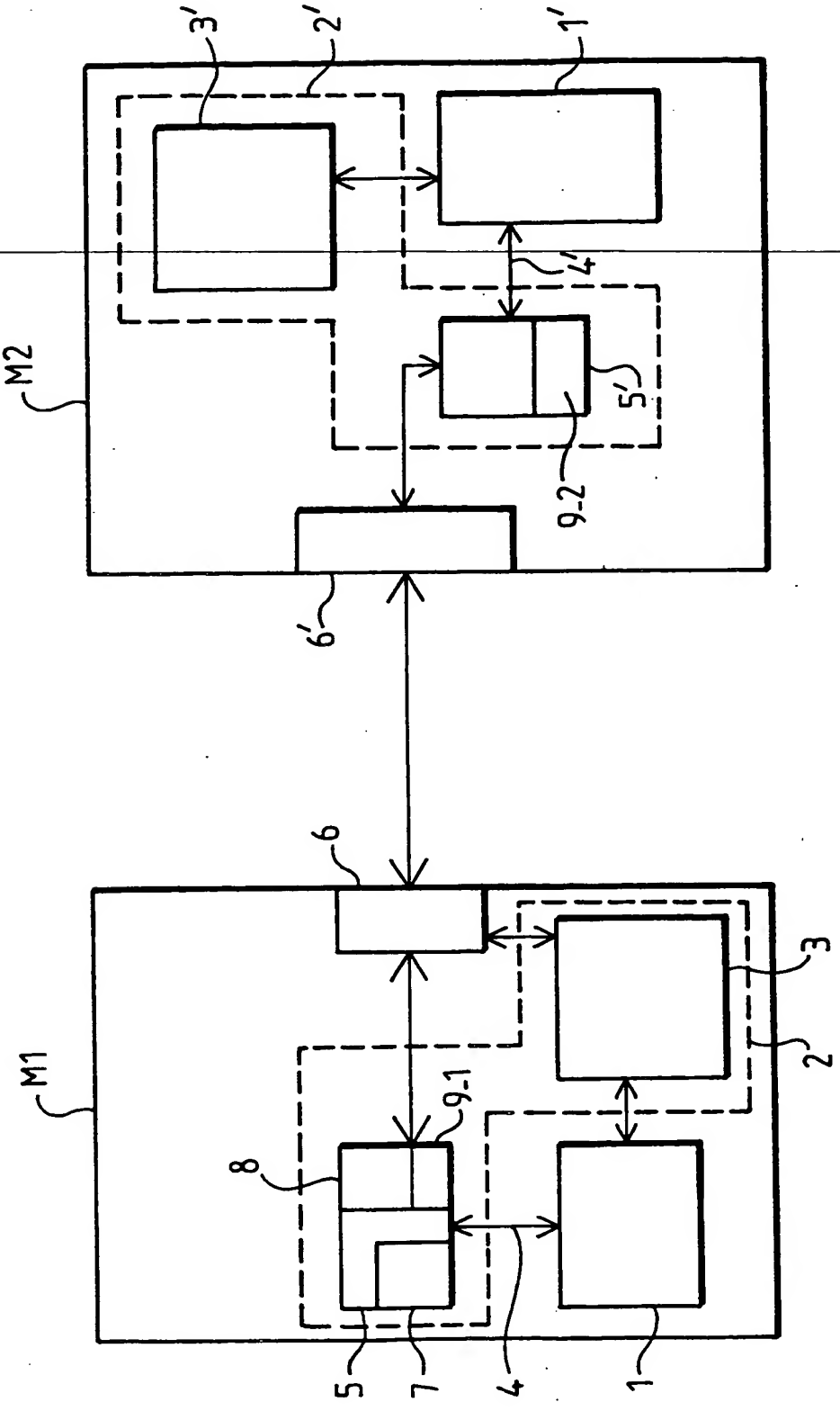

FIG.1

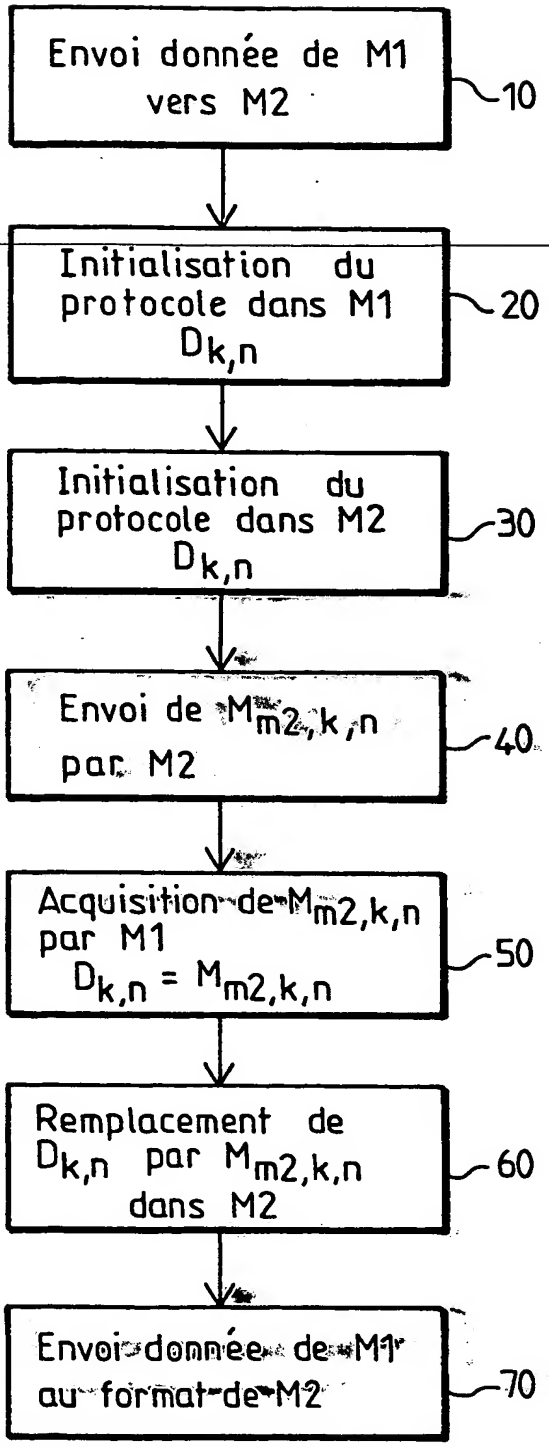
Envoi donnée de M1 vers M2
10
Initialisation du protocole dans M1 $D_{k,n}$
20
Initialisation du protocole dans M2 $D_{k,n}$
30
Envoi de $M_{m2,k,n}$ par M2
40
Acquisition de $M_{m2,k,n}$ par M1 $D_{k,n} = M_{m2,k,n}$
50
Remplacement de $D_{k,n}$ par $M_{m2,k,n}$ dans M2
60
Envoi donnée de M1 au format de M2
70

FIG. 2